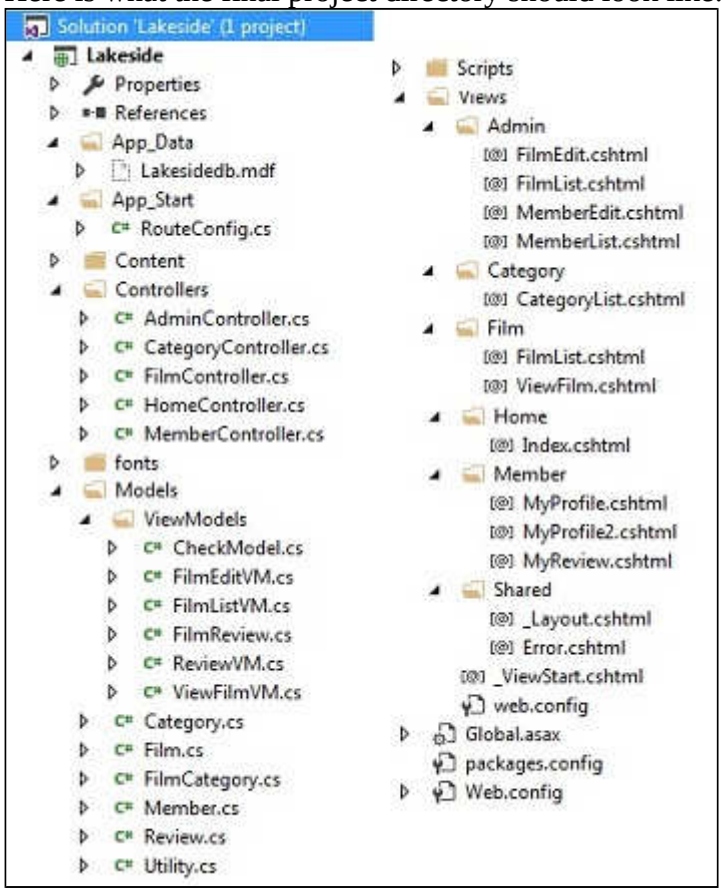
Programming Hints

1. **Directory Structure**
   Here is what the final project directory should look like:



   This structure does not have the account related files to support authentication. Also note that the Categories controller and view will be provided by me as a demonstration of the all-in-one view approach to CRUD.

2. **How to Start**
   - Download and unzip the lakeside_files_04xx17.zip file.
   - Start a new empty project in Visual Studio called Lakeside
   - Create a new database connection called LakesideDB and apply both sql text files.
   - Copy this connection code in the itdev16x_sql_helpers.pdf file to the Web.config file.
   - Add an Home page and Index view that is based upon the layout page option
   - Make sure the Home/Index page is the start page
   - Try it out and make sure the site works
   - Add a new folder called Images inside the Content folder and add a Films folder and members folder within the Images folder. Use the Add existing items menu to "officially" add all the jpg files.
   - Add a new folder within the Models folder called ViewModels
   - Add all the classes within the Models and ViewModels folder using the add existing items menu command.
   - Create new empty controller called Admin
   - Copy all the code within the provided AdminController.cs file into the newly added module.
   - Create 2 views FilmList and FilmEdit and copy the code from each of the provided FilmList.cshtml and FilmEdit.cshtml files.
   - Add this link to the Home/Index page: Edit Films
   - Try out the link and make sure all the functions work
   - You are now on your own now!!

3. **Film List Page**
We will be presenting a dropdownlist of film categories. We will default the selected list item to the first category. All the films in the selected category will be displayed on the screen and the user can click the link on the film he/she wishes to view. Changing the selected dropdownlist item will cause a Postback to occur and the films in the new category will then be displayed.

We will be using a viewmodel class to pass the data to the view. I called the class FilmListVM and here is the class definition:

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.Linq;
 4 using System.Web;
 5 using System.Web.Mvc;
 6
 7 using Lakeside.Models;
 8
 9 namespace Lakeside.Models.ViewModels
10 {
11     public class FilmListVM
12     {
13         public int selectedcatid { get; set; }
14         public List<Film> films { get; set; }
15         public IEnumerable<SelectListItem> catlist { get; set; }
16     }
17 }
```

We need to populate the viewmodel in the FilmList action method of the Films controller. Here is the code to populate it and pass the filled in object to the FilmList view:

```
 1 public ActionResult FilmList(int id=0)
 2 {
 3    FilmListVM filmvm = new FilmListVM();
 4    try
 5    {
 6      dbcon.Open();
 7      filmvm.catlist = Category.GetCategoriesDDList(dbcon);
 8      if (id < 0) filmvm.selectedcatid = id;
 9      else filmvm.selectedcatid = Convert.ToInt32(filmvm.catlist.ToList()[0].Value);
10      filmvm.films = Film.GetFilmListView1(dbcon, filmvm.selectedcatid);
11      dbcon.Close();
12    }
13    catch (Exception ex)
14    {
15      @ViewBag.errormsg = ex.Message;
16      if (dbcon != null && dbcon.State == ConnectionState.Open) dbcon.Close();
17      return View("error");
18    }
19    return View(filmvm);
20 }
21 [HttpPost]
22 public ActionResult FilmList(FormCollection fc)
23 {
24      return RedirectToAction("FilmList", "Film",
25                new { id = Convert.ToInt32(fc["selectedcatid"]) });
26 }
```

Here is a snippet of Razor code showing how to use the dropdownlist box and cause the postback:

```
@model Lakeside.Models.ViewModels.FilmListVM
....
@using (Html.BeginForm("FilmList", "Film", FormMethod.Post))
{
    <text>Choose a Category: </text>
    @Html.DropDownListFor(m =< m.selectedcatid, Model.catlist,
                new { onchange = "document.forms[0].submit();" })
    <br /><br />
....
```

4. **ViewFilm Page**

This page will be run from the link on each film that appears on the FilmList page. The page will retrieve and display all the film bio data along with the additional resources and associated member reviews.

We will be using a viewmodel class that I called ViewFilmVM and here is the class definition for it:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Lakeside.Models;

namespace Lakeside.Models.ViewModels
{
    public class ViewFilmVM
    {
        public Film selectedfilm { get; set; }
        public List<FilmReview> reviewlist { get; set; }
    }
}
```

This model contains all the data in the film table for the selected film as well as all the reviews in a list form.

Here is the ViewFilm action method that will populate the above viewmodel class and call the view:

```
public ActionResult ViewFilm(int id)
{
  ViewFilmVM vm = new ViewFilmVM();
  try
  {
     dbcon.Open();
     vm.selectedfilm = Film.GetFilmSingle(dbcon, id);
     vm.reviewlist = FilmReview.GetFilmReviewList(dbcon, id);
     dbcon.Close();
  }
  catch (Exception ex)
  {
     @ViewBag.errormsg = ex.Message;
     if (dbcon != null && dbcon.State == ConnectionState.Open) dbcon.Close();
     return View("error");
  }
  return View(vm);
}
```

Here is a small snippet of Razor code from the ViewFilm view showing how to use the rating and display the avatar:

```
@foreach (var item in Model.reviewlist)
{
...
   <img src="~/content/images/@(item.Rating.ToString())_star.jpg" /><br />
   <img src="~/content/images/members/@item.Avatar"
                 style="width:100px;height:100px" />
...
}
```

5. **Maintain Members Page**
   Under construction.

6. **MyProfile Page**
   Under construction.

7. **Write a Review Page**
   Under construction.